CS 112  Introduction to Programming

Lecture #2: Java Program Structure
Yang (Richard) Yang

Computer Science Department
Yale University
308A Watson, Phone: 432-6400
Email: yry@cs.yale.edu

Outline
- Admin. and recap
- Java: the programming language
- Programming levels
- Java programming steps
- Java program structure

Admin
- Capping decision to be made by today
- Make sure you sign up on classes v2 server to receive email messages
- Programming Assignment 1 to be posted
  - Due next Wednesday

Recap
- Programming is to apply algorithmic thinking to design computer programs to solve problems
  - Describe each step in a computer language
    - Algorithms represent imperative knowledge vs declarative knowledge
  - Figure out why the computer did not follow the instructions as you expected

Programming Language Choices
## Outline

- Admin. and recap
  - Java: the programming language

---

## Java Programming Language: Key Designers

- **Bill Joy**
  - BSD Unix guy from UC Berkeley
  - co-founder of Sun Microsystems (1982)
  - focus on "the network is the computer" workstation
  - failure: focusing on network was ahead of its time, but missed the boat on PC revolution

- **James Gosling**
  - early fame as the author of "Gosling Emacs"
  - killed by open GNU emacs
  - then onto Sun's "NeWS" windows system
  - failure: keeping things proprietary led to "kiss of death"

---

## Java Programming Language: History

- Joy and Gosling joined force: Sun subsidiary, FirstPerson, Inc. (1992)
  - target consumer electronics: PDAs, appliances, phones, all with cheap infra-red kinds of networks
  - need a language that's safe, portable, secure, wired
    - started working on C++
    - soon gave up hope, decided to start from scratch
  - a little ahead of time (again): PDAs died with the demise of Apple Newton
  - switched to interactive TV (ITV)
    - the resulting language was called "Oak"
  - a little ahead of time (yet again): ITV died too

- Third time's the charm
  - the Web exploded
  - Oak became Java

---

## Java Features

- **Java is a modern, elegant, object-oriented programming language**
  - simpler than other object-oriented languages [e.g., C++]
  - Java is the basis of other modern programming languages [e.g., Microsoft C#]

- **Java is (largely) portable --- write once run everywhere**
  - Java supports multiple platforms (Unix, Windows, Mac), multiple types of devices (desktops, phones, embedded devices)

- **Java has rich libraries and good support**
  - good multimedia, graphics packages
  - good client-server and network support
  - good, free Integrated Development Environments (IDE)
### Java is Still Evolving

<table>
<thead>
<tr>
<th>Version</th>
<th>Year</th>
<th>Important New Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1996</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>1997</td>
<td>Inner classes</td>
</tr>
<tr>
<td>1.2</td>
<td>1998</td>
<td>Swing, Collections</td>
</tr>
<tr>
<td>1.3</td>
<td>2000</td>
<td>Performance enhancements</td>
</tr>
<tr>
<td>1.4</td>
<td>2002</td>
<td>Assertions, XML</td>
</tr>
<tr>
<td>5</td>
<td>2004</td>
<td>Generic classes, enhanced for loop, auto-boxing, enumerations</td>
</tr>
<tr>
<td>6</td>
<td>2006</td>
<td>Library improvements</td>
</tr>
<tr>
<td>7</td>
<td>2011</td>
<td>Lambda</td>
</tr>
<tr>
<td>8</td>
<td>2014</td>
<td>More Lambda, Javascript runtime</td>
</tr>
</tbody>
</table>

- New features added by following the Java Community Process
- Others extend Java to other settings: Google Android uses Java on mobile devices

### Machine Language

- The "brain" of a computer is its Central Processing Unit (CPU)
- A CPU can understand only very basic instructions
  - e.g., store a given value at a memory location; do some arithmetic operations; compare two values; start to execute the instruction at another location
- The instruction set of a CPU forms the machine language of the CPU
- Different machines understand different machine languages

### High-Level Programming Languages

- A high-level programming language enables a programmer to specify, in a high level (close to natural language), what data a computer will act upon, how these data will be stored, and what actions to take under various circumstances
- The syntax and grammar of a high-level language is independent of CPU

### High-Level Languages

### Problem

- **Language barrier**
  - Computers: understand machine platform languages—-to build efficient hardware
  - Programmers: want more readable high-level languages—-to be more productive
A program written in a high-level language must be translated into the language of a particular platform (type of CPU and operating system) before execution.

A compiler is a program which translates source code into a specific target platform (CPU + OS).

Multiple versions of the same software

High-level Picture

To be platform independent, Java designers introduced Java Virtual Machine (JVM), a machine different from any physical platform, but a virtual machine.

A Java compiler translates Java source code (.java files) into bytecode (in .class files).

Java Virtual Machine

To execute a Java program, another piece of software called an interpreter, translates between bytecode and the actual machine.

Java Translation and Execution
Comparing Traditional (e.g., C/C++) and Java Software Development

Traditional, e.g., C/C++
- A developer writes a program in C/C++
- The C/C++ source code is generally considered proprietary and not released
- The developer compiles the C/C++ program for each platform it intends to support and distributes one version for each platform
  - thus each program has multiple compiled versions
  - each compiled version can run by itself
- Platform dependency handled by each software developer

Java
- A developer writes a program in Java
- The Java source code is generally considered proprietary and not released
- The developer compiles the Java program to bytecode and distributes the bytecode version
  - thus each program has only one compiled version
  - the compiled bytecode needs an interpreter for each platform
- Platform dependency handled by platform vendor

High-level Picture

Outline

- Admin. and recap
- Java: the programming language
- Programming levels
- Java programming steps

Recall: Java Programming Steps

- Programming in Java consists of 3 simple steps
  - Create and edit “Java source code” (.java files)
  - Compile into “Java bytecode” (.class files)
  - Execute bytecode with a “Java interpreter”

Programming in Java (Step 1): Create/Edit

- The basic way is to use a text editor
  - Example editors: Notepad, TextEdit, emacs, vim, etc.
  - Note: MS Word is NOT a text editor
- The key is that your Java file cannot include any markup or stylistic formatting; just text.
- You enter your Java code following Java Language syntax.

First Java Program

```
/***************************************************************************/
/* Prints "Hello World" */
/* Everyone’s first Java program. */
/***************************************************************************/
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}```
Programming in Java (Step 2): Compile

- Compile a Java program
  \$ javac HelloWorld.java

- Take a look to see that HelloWorld.class is generated
  \$ ls
  HelloWorld.java HelloWorld.class

Programming in Java (Step 3): Execute

- Run Java interpreter
  \$ java HelloWorld

Another Java Program

```java
public class Hello2 {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
        System.out.println();
        System.out.println("This program produces");
        System.out.println("four lines of output");
    }
}
```

Programming in Java: Method 2

- Another way is to use an Integrated Development Environment (IDE)
  - Example IDEs: Eclipse, DrJava, etc.
  - An IDE usually presents the user with a space for text (like an editor) but layers additional features on top of the text for the user’s benefit.
    - Note: The underlying file contains pure text, just like a text editor.
  - These features can be very useful and save time.
    - Example features are GUI compile, GUI execution, code completion, and syntax highlighting.
  - IDEs take more time to get started than a simple text editor, e.g.,
    - set up where to find the “java” and “javac” programs
    - find out where does the IDE save my file

Outline

- Admin. and recap
- Java: the programming language
- Programming levels
- Java programming steps
- Java program structure

Syntax and Semantics

- The syntax rules of a language define how we can put characters together to make a valid program
- The semantics of a program define what a program does
  - a program that is syntactically correct is not necessarily logically (semantically) correct
  - This is similar in natural language, e.g.,
    - "Yale University has no dining halls."
    - "Harvard can beat Yale."
- At the very beginning, the challenge is to resolve syntax issues; but quickly, we will focus on the semantics—let a program do what we want
### Top-Down Syntax Structure of a Java Program

A class:
- has a name, defined in a file with same name
- starts with {, and ends with }
- includes a group of methods

Public class `class name` {
    public static void main(String[] args) {
        <statement>
        <statement>
        ...
        <statement>
    }
}

A method:
- has a name
- starts with {, and ends with }
- includes a group of statements

Statement:
- a command to be executed
- end with ;

### System.out.println statement

- A statement that prints a line of output on the console.
  - pronounced "print-linn"

- Two ways to use `System.out.println`:
  - `System.out.println(<string>);`
    - Prints the given message `<string>` as output.
  - `System.out.println();`
    - Prints a blank line of output.

### Syntax: Strings

- `string`: A sequence of characters that starts and ends with a " (quotation mark character).
  - The quotes do not appear in the output.
  - Examples:
    - "hello"
    - "This is a string. It's very long!"

- Restrictions:
  - May not span multiple lines
    - "This is not a legal String."

### Examples

- Which of the following are legal strings in Java?
  - "This is a string. It's very long!"
  - "This cool string spans two lines."
  - "It is a great thing when children cry, "I want my mommy"!"

### Escape Sequences

- `escape sequence`: A special sequence of characters used to represent certain special characters in a string.
  - `\t`: tab character
  - `\n`: new line character
  - `\"`: quotation mark character
  - `\\`: backslash character

- Example:
  - `System.out.println("Hello how are you?\\\n")`;
  - Output:
    - `Hello how are "you"?\n`

### Practice Slides (Out of Class)
Questions

1. What is the output of the following `println` statements?
   ```java
   System.out.println("ta\htoc");
   System.out.println("\\");
   System.out.println("\n\n\n");
   System.out.println("C:\\in\the\downward\spiral");
   ```

2. Write a `println` statement to produce this output:
   ```text
   /
   / /
   / / /
   / / / 
   / / / / 
   / / / / / 
   ```

Answers

1. **Output of each `println` statement:**
   ```text
   a       b       c
   \      
   '      '''
   C:
   in
   he downward spiral
   ```

2. A `println` statement to produce the line of output:
   ```java
   System.out.println("/ \ // \ /// \ \/// \ \\/// \ /// ");
   ```

Questions

1. What `println` statements will generate this output?
   ```java
   System.out.println("This program prints a quote from the Gettysburg Address.");
   System.out.println("Four score and seven years ago, our "fore fathers" brought forth on this continent a new nation.");
   System.out.println("This quote is from Irish poet Oscar Wilde: ");
   System.out.println("Music makes one feel so romantic - at least it always gets on one's nerves - which is the same thing nowadays.");
   ```

2. A "quoted" String is "much" better if you learn the rules of "escape sequences."
   Also, "" represents an empty String.
   Don't forget: use \" instead of "!
   '' is not the same as ".

Answers

1. **println statements to generate the output:**
   ```java
   System.out.println("This program prints a");
   System.out.println("quote from the Gettysburg Address.");
   System.out.println("Four score and seven years ago, our "fore fathers" brought forth on this continent a new nation.");
   System.out.println("This quote is from Irish poet Oscar Wilde: ");
   System.out.println("Music makes one feel so romantic - at least it always gets on one's nerves - which is the same thing nowadays.");
   System.out.println("A "quoted" String is "much" better if you learn the rules of "escape sequences."
   Also, "" represents an empty String.
   Don't forget: use \" instead of "!
   '' is not the same as \".
   ```

2. A "quoted" String is "much" better if you learn the rules of "escape sequences."
   Also, "" represents an empty String.
   Don't forget: use \" instead of "!
   '' is not the same as \".
Backup Slides

Assembly Languages

- Assembly language or simply assembly is a human-readable notation for the machine language. It's much easier to remember:
  ```assembly
  movl %al, 97
  ``
  ```binary
  10100000 01100001
  ```

Example assembly code fragment.

Some Major Types of High-Level Languages

- **Procedural languages**: programs are a series of commands
  - Pascal (1970): designed for education
  - C (1972): low-level operating systems and device drivers
- **Functional programming**: functions map inputs to outputs
- **Object-oriented languages**: programs use interacting "objects"
  - Smalltalk (1980): first major object-oriented language
  - C++ (1985): "object-oriented" improvements to C
    - successful in industry; used to build major OSes such as Windows
  - Java (1995): designed for embedded systems, web apps/servers
    - runs on many platforms (Windows, Mac, Linux, cell phones...)

Some Major Types of High-Level Languages

- **Procedural languages**: programs are a series of commands
  - Pascal (1970): designed for education
  - C (1972): low-level operating systems and device drivers
- **Functional programming**: functions map inputs to outputs
- **Object-oriented languages**: programs use interacting "objects"
  - Smalltalk (1980): first major object-oriented language
  - C++ (1985): "object-oriented" improvements to C
    - successful in industry; used to build major OSes such as Windows
  - Java (1995): designed for embedded systems, web apps/servers
    - runs on many platforms (Windows, Mac, Linux, cell phones...)